

## FUNCTIONAL ANALYSIS (Data Flow Diagrams)

The functional view of the system is represented by Data Flow Diagrams (DFDs). The DFD is a simple diagrammatic representation showing how data flows through a system, its path through or between various processes and files (data stores) in the system.

The main advantages are:

- the boundary, or scope of the system under investigation is clearly defined
- the movement and content of data is shown
- a 'top-down' or hierarchical decomposition allows progression from the broad view to the final detailed representation at the lowest required level.
- they use simple, limited conventions so are easy to understand. Hence they are an acceptable communication medium to both the project team and users.

There are five symbols used in the drawing of a DFD.

- 1 **PROCESS:** The process name should show the actual process data goes through and should be meaningful, ie not showing the department or function where the PROCESS is carried out. This is also sometimes called the Transform Centre.

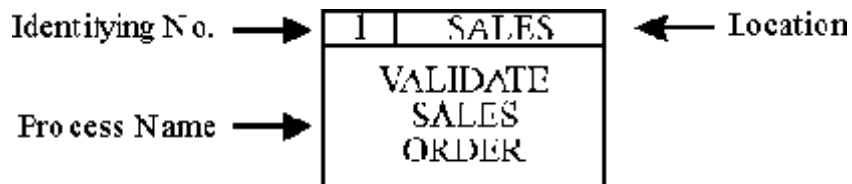


Figure 1

- 2 **DATA FLOW:** Shows the movement of data from one point to another. Must also show the data structure name. This should be a unique name which describes the data content.



Figure 2

- 3 **DATA STORE:** Represents data at 'rest' ie a file or where data is stored for later use or retrieval. Store name should be meaningful.



Figure 3

- 4 **SOURCE/DESTINATION** (or SINK): Shows the Source of data into the system and the Destination of data from the system. Also known as Source/Sink or External Entity



Figure 4

- 5 **PHYSICAL FLOWS:** Shows the passage of physical items such as GOODS through a system. This should only be used for clarity i.e. when the physical resource is the only means of communicating data.



Figure 5

In order to keep the diagrams clear and simple to follow, and to avoid data flows crossing each other, the symbols for data stores and external entities (sources and destinations) can be repeated for the same store or entity. This is shown by an asterisk in the top right-hand corner of the symbol.

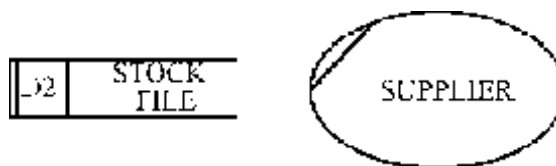


Figure 6

### Basic DFD Conventions

- There should be 5 (+/- 2) processes to each DFD
- Processes should be numbered sequential, not showing order of operation.
- Data Stores can be duplicated, in which case an asterisk is shown in the top right-hand corner.
- External Entities can be duplicated, in which case an asterisk is shown in the top right-hand corner.
- Exception/Error processing is not included at this stage unless it is a significant aspect of the system's operation.

DFDs are constructed hierarchically, each level being decomposed to show more detail until the lowest level process is shown.

Hierarchical models enable the Analyst to maintain several levels of detail in the constructed models. A high-level model will have little detail on it since its primary purpose is to highlight the most important features of the system. A motorway map will only show the motorway network and/or most of the main roads linking the towns and cities. It will be sufficient to plan a journey from, say, Bristol to Nottingham, but it will not help the motorist find his way to a particular district of Nottingham. For this purpose a lower level model is required showing the main trunk roads, and this should permit the general location of the area required. At this point a still lower level model is needed if the motorist is to find the actual street. He will then consult a street plan, and possibly a location map of the particular premises he wishes to visit to complete his journey.

A similar model is required in information systems design.

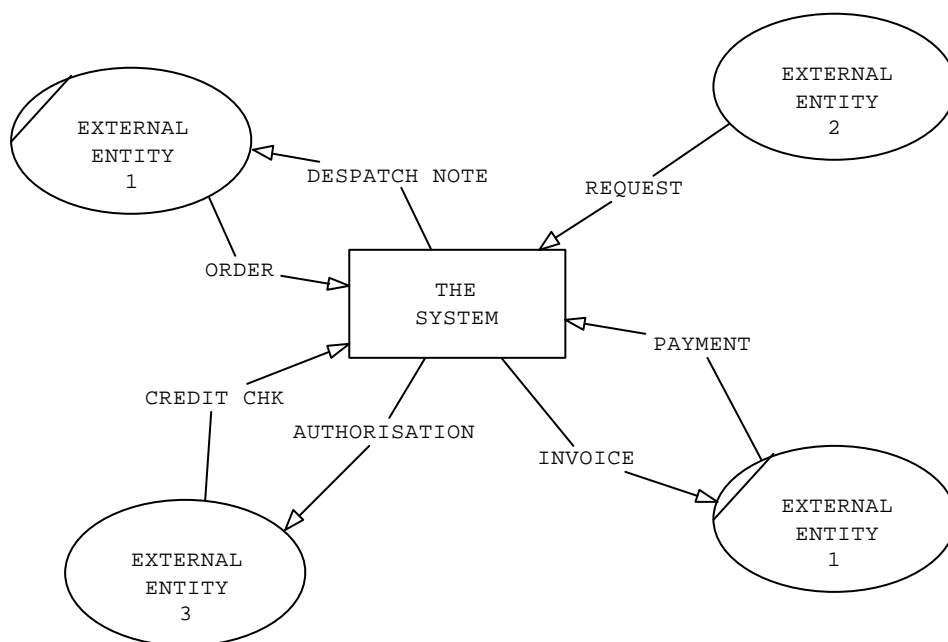
## Guidelines for Production of Data Flow Diagrams

### Top Level

Identify the system boundary. Identify data flows which cross this boundary. Identify external entities which are the sources or destinations of these data flows. Remember that the external entities can be outside the organisation eg Customer, or inside the organisation but outside the system boundary eg Accounts.

These data flows will be the inputs and outputs for the total system and put the system into context with the 'real world', hence it is called a CONTEXT DIAGRAM, also known as a LEVEL 0 DFD.

The Context Diagram will lead naturally into the Level 1 DFD. The major business functions (processes) within the system will have been identified during the investigation stage. It is now necessary to decide which of these processes handle the identified inputs and produce the outputs.



**Figure 7 Context Diagram**

Draw a large rectangle on the page to represent the system boundary, leaving a large margin to accept the external entities. Take each external entity in turn, and using the data flows ask the question:

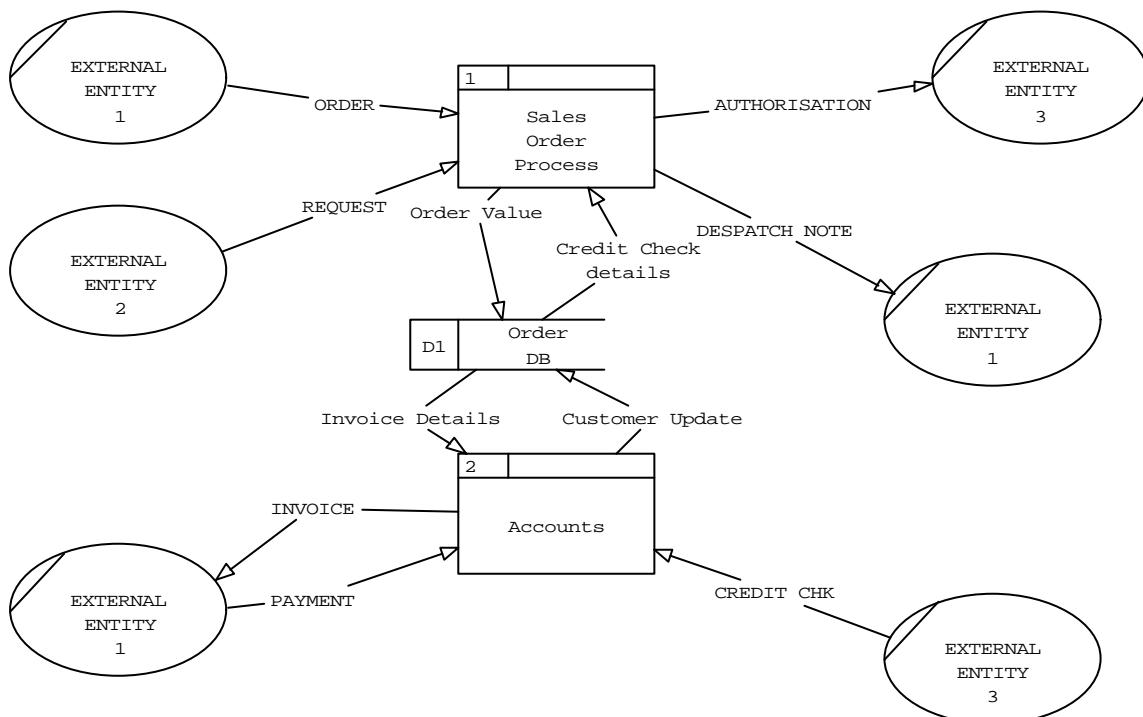
"Which high level process handles the input data flow(s)?"

and

"Which process produces the output data flow(s)?"

Draw these process boxes inside the perimeter of the rectangle but towards the outer edge.

Add the external entities and the data flows and this will provide a skeleton data flow diagram (Figure 8). At this point it is essential to check that all data flows on the Context Diagram have been included on the skeleton DFD. Internal processes, data stores and their connecting data flows now have to be added. Data stores are only represented when they are accessed by more than one process.



**Figure 8**

If in doubt, one way to tackle this next stage is to take each output data flow in turn and determine the data required to perform the process which creates that output. Usually this will be a data store (or data stores) which can be added. Then questions such as:

"Which processes create/update these data stores?"

will provide data flows from existing processes on the diagram or highlight the need for additional internal processes. In this way the total model can be built up.

## Decomposing or Levelling DFDS

Using the same rules and questions as above, each process on the top level (Level 1) DFD can be expanded (Levelled) to a maximum of 5 (+/- 2) processes on a Level 2 DFD. If necessary, those Level 2 processes can be expanded to Level 3. In most systems two or three levels are sufficient, although with highly complex processing further levelling may be necessary.

All communications (data flows) between the process being levelled and external entities, data stores and the processes must be maintained. Any 'local' data stores (ie those files only used by that process) will be added and represented inside the rectangle. Error and exception handling routines must also be included (Figure 9).

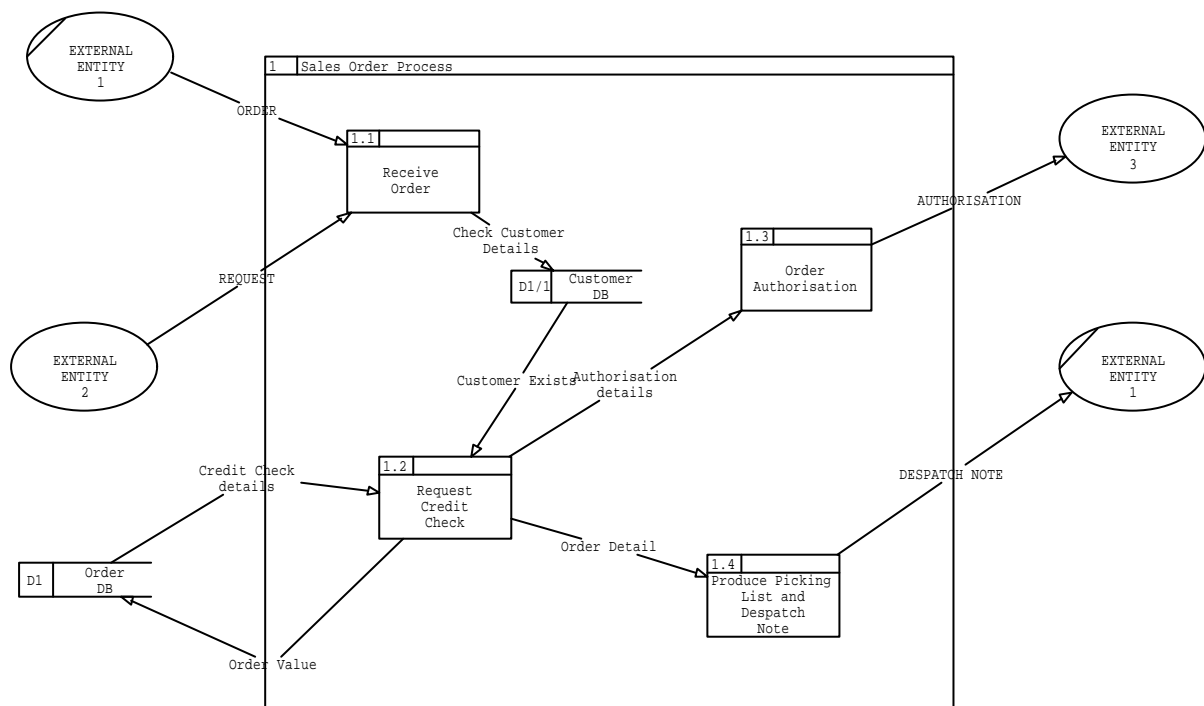


Figure 9

## DFD Numbering Conventions

Functions: Level 1: Sequentially 1, 2, 3, etc.  
 Level 2: 1.1, 1.2, 1.3, etc.  
 Level 3: 1.1.1, 1.1.2, 1.1.3, etc.

Data Stores: D - Computer Files: D1, D2, D3, etc.  
 M - Manual Files: M1, M2, M3, etc.  
 T - Transient or  
 Temporary Files: T1, T2, T3, etc.

Any data stores added at lower levels will be allocated the next available sequential number.

## Supporting Documentation

- Procedure Specification (Mini-spec): A description of the lowest level process identified on the lower level DFD. Can be entered as Text, Decision Table, Decision Tree, Flowchart, Structured English or Pseudo Code.
- Data Flow Description: Identifier and the major data items represented by the flow.
- Data Store Description: Identifying data item ie, the Key. Major data items within the file.

## Use of DFDs in Structured Methodology

- PHYSICAL:** First use of DFDs. Used to record the Current system. How it operates and what it does now. Refers to current organisation, equipment used, actual files, actual process and job functions.
- LOGICAL:** Conceptual or logical representation of how the Current system could operate if all physical constraints were removed.
- REQUIRED:** The logical representation of the New system amended to include the requirements and solve the problems identified by the users and the project team and listed in the Terms of Reference.

## Logicalisation

Having removed all physical constraints to arrive at the Current logical system, the New logical system can be defined by considering the following when adding user requirements.

- i) Look for redundant Data Flows, Data Stores and Processes and remove them.
- ii) Where necessary rationalise Data Stores by either combining with other common data stores or by separating when the data will be accessed in different ways.
- iii) Ensure that all physical reference to the Systems boundaries or problems are removed.
- iv) Replace all Process, Data Flow and Data Store names with logical names, describing the actual process or Data in functional terms not user jargon or departmental terminology.
- v) Ensure that each Process only passes on the data required.